1.单选题（1.5分）

windows操作系统中下列哪个是复制操作()

A.control + C　B.control + V　C.control + P　D.control + A

2.单选题（2分）

下列一段代码是什么排序()

```
void sort_A(int* a, int n) {
  for (int i = 2; i <= n; ++i) {
    int key = a[i];
    int j = i - 1;
    while (j > 0 && a[j] > key) {
      a[j + 1] = a[j];
      --j;
    }
    a[j + 1] = key;
  }
}
```

A.冒泡排序　B.插入排序　C.选择排序　D.计数排序

3.单选题（2分）

欧拉函数（Euler's totient function) $\varphi(x)$表示的是小于等于x和x互质的数的个数。

那么假设x = 10007，求$\varphi(x)$（ ）。

A.1

B.0

C.1005

D.10006

4.单选题（2分）

表达式a* (b+c)* d +e* f的后缀表达式为（ ），其中“*”和“+”是运算符。

A.abc+*d *ef *+

B.abc+*d *ef + *

C.abc+*d +ef *

D.+***+bcadef

5.单选题（1.5分）

众所周知林老师很喜欢"人一我十，人百我万"这句话，现在他想要在C++中输入这句话，他希望能够在计算机中显示出这句话，以下哪个选项错误( ).

A. string a; cin >> a; cout << a ; 能够正确输出

B.char b[110]; cin >> b; cout << b； 能够正确输出

C.char b[110]; scanf("%s",b); printf("%s\n",b);能够正确输出

D.char b[110]; int len = strlen(b); for(int i = 0; i < len; i++){

printf("%c",b[i]);

} 不能够正确输出

6.单选题（1.5分）

UDP 属于什么层( )。

A. 传输层

B. 网络层

C. 物理层

D. 应用层

7.单选题（2分）

5个小朋友站成一个环，其中有两个双胞胎，如果要求这两个双胞胎必须相邻，则有（ ）种不同排列方法。

A. 12

B. 36

C. 24

D. 48

8.单选题（2分）

对于有n(n >= 3)个顶点、n * (n - 1) / 2条边的无向图，每条边的权值为1，没有自环，没有重边，每走过一条边就会获得该边的权值，请问有多少种路径从点1走到点n并且获得的权值为3()。

A. $n^3 - 4 * n^2 + 6 * n - 4$

B. n * (n + 1)

C. 5 * 4^{n-2}

D.无法确定

9.单选题（1.5分）

如果一棵二叉树的中序遍历是 DBFEGAC，那么它的先序遍历可能是( )。

A. ABDEFGC

B. ACBDEFG

C. AEGFBDC

D. AEBFDGC

10.单选题（2分）

若两条直线相交则会有一个交点，那么五条直线相交，最多能够有（）个交点。

A.5

B.10

C.12

D.15

11.单选题（1.5分）

在一个有向图中，所有顶点的入度之和等于所有顶点的出度之和的（ ）倍。

A.1/2

B.1

C.2

D.4

12.单选题（2分）

由3个a，5个b和2个c构成的所有字符串中，包含子串“abc”的共有（ ）个。

A. 40320 B. 39600 C. 840 D. 780

13.单选题（1.5分）

如果根的高度为 1,具有 61 个结点的二叉树的高度可能为()。

A.63

B.62

C.5

D.10

14.单选题（2分）

设 x=true, y=true, z=false，以下逻辑运算表达式值为真的是().

A. (y ∨ z) ∧ x ∧ z

B. x ∧ (z ∨ y) ∧z

C. (x ∧ y) ∧ z

D. (x ∧ y) ∨ (z ∨ x)

15.单选题（2分）

以下哪一个不是栈的基本运算( )。

A.删除栈顶元素

B.删除栈底的元素

C.判断栈是否为空

D.将栈置为空栈

16.填空题(12分)

```
01  #include<iostream>
02
03  using namespace std;
04
05  const int p = 1000007;
06  onst int N = 110;
07  int a[N],dp[N][N];
08  int main(){
09      int n,m;
10      cin >> n >> m;
11      for(int i = 1; i <= n; i++){
```

```
12          cin >> a[i];
13      }
14      for(int i = 0; i <= a[1]; i++){
15          dp[1][i] = 1;
16      }
17      for(int i = 1; i <= n; i++){
18          dp[i][0] = 1;
19      }
20      for(int i = 2; i <= n; i++){
21          for(int k = 1; k <= m; k++){
22              for(int l = 0; l <= min(a[i],k); l++){
23                  dp[i][k] = (dp[i][k] + dp[i - 1][k - l]) % p;
24              }
25          }
26      }
27      cout << dp[n][m] << endl;
28      return 0;
29  }
```

•判断题（1.5分一个)

1）如果删除 14 15 16行对答案没有影响()

2）若将22行改成for(int l = min(a[i],k); l >= 0; l--) {对答案没有影响()

3）若将21行改成for(int k = m; k >= 1; k--){ 对答案没有影响（)

4）若把 23行 %p去掉答案也不会超过int范围()

•选择题（2分一个)

5）若输入

```
2 4
3 2
```

答案为()

A.1　B.2　C.3　D.4

6）若输入

```
3 4
1 2 3
```

答案为()

A.2　B.3　C.4　D.5

17.填空题

优先队列是普通的队列是一种先进先出的数据结构，元素在队列尾追加，而从队列头删除。在优先队列中，元素被赋予优先级。当访问元素时，具有最高优先级的元素最先删除。优先队列具有最高级先出（first in, largest out）的行为特征。通常采用堆数据结构来实现。

```
01  #include <bits/stdc++.h>
```

```
02  using namespace std;
03  const int maxn = 110;
04  int n, k, w[maxn]; double v[maxn];
05  struct node {
06      double pt; int total;
07      bool operator < (const node &temp) const {
08          return pt < temp.pt;
09      }
10  };
11  priority_queue<node> pq;
12  int main() {
13      cin >> n >> k; double ans = 0;
14      for(int i = 1;i <= n; ++i) cin >> v[i];
15      for(int i = 1;i <= n; ++i) cin >> w[i];
16      for(int i = 1;i <= n; ++i)
17          pq.push({v[i] / w[i], w[i]});
18      while(k > 0 && pq.size()) {
19          node u = pq.top(); pq.pop();
20          ans += min(k, u.total) * u.pt;
21          k -= min(k, u.total);
22      }
23      cout << ceil(ans) << '\n';
24      return 0;
25  }
```

假设所有输入的数据均为[0, 100]之间的自然数，回答下列问题。

l 判断题

1. 存在可以被构造出的输入数据，使得程序无法正确运行。（ ）
2. 对于优先队列pq，当其非空时，若取出pq中的top处的元素，该元素必然是优先队列pq中pt属性较小的元素。（ ）
3. 当程序正常运行结束时，ans一定小于等于$\sum v[i](1 \leq i \leq n)$。（ ）

4. 把07 08 09行替换成

```
bool friend operator < (node a,node b){
    return a.pt < b.pt;
}
```

对程序没有影响()

l 选择题

4. 当n = 5, k = 7, 数组v[i]为[1, 2, 3, 4, 5], w[i]为[5, 4, 3, 2, 1]时，程序的输出结果为（ ）

A. 13　　B. 2

C. 12　　D. 1

5.当n = 5, k = 3, 数组v[i]为[1, 5, 3, 4, 2], w[i]为[2, 5,1, 4, 3]时，程序的输出结果为（ ）

A. 3　　B. 4

C. 5　　D. 6

```
01  #include <bits/stdc++.h>
02  using namespace std;
03  int main() {
04      int cnt[50]; memset(cnt, 0, sizeof cnt);
05      string str; cin >> str;
06      for(int i = 0;i < str.size(); ++i)
07          cnt[str[i] - 'a']++;
08      int ans = 0; bool judge = false;
09      for(int i = 0;i <= 25; ++i) {
10          if(cnt[i] % 2 == 0) ans += cnt[i];
11          else ans += cnt[i] - 1, judge = true;
12      }
13      if(judge) ++ans;
14      cout << ans << endl;
15      return 0;
16  }
```

假设输入的字符串仅包含小写字母，且长度不超过105，完成下列问题。

l 判断题

5. 将第09行的i <= 25改为i <= 30，不影响程序的结果。（ ）

6. 当第13行中的judge 为true时，求出的ans为偶数。（ ）

l 选择题

7. 当输入的字符串为helloworld时，输出的答案为（ ）

A. 7　　B. 5

C. 8　　D. 6

8. 当输入的字符串为woshidashazi时，输出的答案是( )

A.7　　B.8

C.9　　D.10

一、 完善程序（单选题，每小题3分，共计30分）

（一） 现在要求你求出满足下列要求的序列的数量，答案对109+7取模。

① $a_0 = 0$

② $\forall\, i \in [1, n]$，满足 $a_i = a_{i-1} + x$ 或 $a_i = a_{i-1} + y$

③ $\forall\, i \in [1, n]$，满足 $a_i \bmod p \neq 0$

输入数据格式为第一行一个正整数t$(1 \le t \le 1000)$，表示有t组询问。

对每组询问给出一行四个正整数n, p, x, y，以空格分开。

保证$1 \le \sum n \le 10^4, 1 \le p, x, y \le 10^9$。对每组询问你都需要回答一行正整数表示答案。

```
#include < cstdio >
#include < iostream >
#include < algorithm >
#include < cmath >
#include < cstring >
using namespace std;
const int maxn = 1e4 + 10;
const int mod = 1e9 + 7;
int n, dp[2 * maxn];
long long p, x, y;
int main( ) {
  int t; cin >> t;
  while(t--) {
    int ans = 0;
    cin >> n >> p >> x >> y;
    if(x == y) {
```

```
    bool judge = true;
    for(int i = 1;i <= n; ++i) if(_①_) judge = false;
    cout << _②_ << '\n';
    continue;
   }
   dp[maxn] = 1;
   for(int i = 1;i <= n; ++i) {
    for(int j = 0;j <= i; ++j) {
     int k = _③_; dp[maxn + k] = 0;
     if(_④_)
      dp[maxn + k] = (_⑤_) % mod;
     if(i == n) ans = (ans + dp[maxn + k]) % mod;
    }
   }
   cout << ans << '\n';
   for(int i = -n; i <= n; ++i) dp[maxn + i] = 0;
  }
  return 0;
}
```

1. ①处应填（ ）

A. x % p == 0　　B. x * i % p == 0

C. i % p == 0　　D. judge && (x % p == 0)

2. ②处应填（ ）

A. judge ? 1 : 0　　B. judge ? n : 0

C. judge ? 0 : 1　　D. judge ? 0 : n

3. ③处应填（ ）

A. j　　B. j + (i – j)

C. i - j　　D. j – (i – j)

4. ④处应填（ ）

A. (x * i + y * j) % p == 0

B. (x * i + y * j) % p != 0

C. (x * j + y * (i – j)) % p == 0

D. (x * j + y * (i – j)) % p != 0

5. ⑤处应填（ ）

A. dp[maxn + k + 1] + dp[maxn + k - 1]

B. dp[maxn + k + 1] - dp[maxn + k - 1]

C. dp[k + 1] + dp[k - 1]

D. dp[k + 1] - dp[k - 1]

（二） 现在你有一个包含n个不同的字符串的串集S。现在询问你有多少个不同的长度为m的字符串，使得字符串中至少有一个连续子串，位于串集S中。由于答案可能很大，你需要对10007取模。

所有出现的字符均只考虑大写字母。

数据范围有$1 \leq n \leq 60, 1 \leq m \leq 100$，每个字符串的串长$\in [1, 100]$

输入格式为第一行两个正整数n,m，表示你拥有的串集大小为n，你可以构造的字符串的长度为m。

接下来n行，每行一个字符串，表示串集中的一个字符串。

提示：考虑用所有字符串的数量减去非法字符串的数量。

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 6000 + 10;
const int mod = 10007;
struct Aho_corasick{
 int ch[maxn][30], f[maxn], tot;
   bool endp[maxn];
   Aho_corasick() {
      memset(ch, 0, sizeof ch);
      memset(endp, 0, sizeof endp);
   memset(f, 0, sizeof f);
      tot = 0;
   }
   void insert(string &s) {
      int n = s.size(); int u = 0;
      for(int i = 0;i < n; ++i) {
         if(!ch[u][s[i] -' A']) 
            ch[u][s[i] -' A'] = ++tot;
         u = ch[u][s[i] -' A'];
      }
      _①_;
   }
```

```
    void getfail() {
        queue< int > q;
        for(int i = 0;i <= 25; ++i)
            if(ch[0][i])
                q.push(ch[0][i]);
        while(!q.empty()) {
            int u = q.front(); q.pop();
            for(int i = 0;i <= 25; ++i) {
                int v = ch[u][i];
                if(v) {
                    f[v] = ch[f[u]][i];
          _②_
                    q.push(v);
                } else {
                    ch[u][i] = ch[f[u]][i];
                }
            }
        }
    }
} AC;
int quickpow(int x, int y) {
  int base = x, ans = 1;
  while(y) {
    if(y & 1) _③_
    base *= base;
    ans %= mod; base %= mod;
    y /= 2;
  }
  return ans;
}
int ans, dp[105][maxn];
int main() {
  int n, m; cin >> n >> m;
  ans = quickpow(26, m);
```

```
  for(int i = 1;i <= n; ++i) {
    string str; cin >> str;
    AC.insert(str);
  }
  AC.getfail();
  int max_state = AC.tot;
  dp[0][0] = 1;
  for(int i = 0;i < m; ++i) {
    for(int state = 0;state <= max_state; ++state) {
      for(int nxt = 0;nxt <= 25; ++nxt) {
        if(_④_) continue;
        _⑤_ += dp[i][state];
        _⑤（两处⑤所填内容相同）_ %= mod;
      }
    }
  }
  for(int state = 0; state <= max_state; ++state)
    ans = (ans + mod - dp[m][state]) % mod;
  cout << ans << '\n';
  return 0;
}
```

6. ①处应填（ ）

A. endp[u] = 0　　B. endp[u] = 1

C. f[u] = 0　　D. f[u] = 1

7. ②处应填（ ）

A. if(endp[f[u]]) endp[u] = 1;

B. if(endp[f[u]]) endp[v] = 1;

C. if(endp[f[v]]) endp[u] = 1;

D. if(endp[f[v]]) endp[v] = 1;

8. ③处应填（ ）

A. ans *= base;　　B. base *= ans

C. ans += base　　D. base += ans

9. ④处应填（ ）

$A.\ AC.f[AC.ch[state][nxt]]$

$B.\ AC.endp[AC.ch[state][nxt]]$

$C.\ AC.f[AC.ch[nxt][state]]$

$D.\ AC.endp[AC.ch[nxt][state]]$

10. ⑤处应填（ ）

$A.\ dp[i][AC.f[AC.ch[state][nxt]]]$

$B.\ dp[i+1][AC.f[AC.ch[state][nxt]]]$

$C.\ dp[i][AC.ch[state][nxt]]$

$D.\ dp[i+1][AC.ch[state][nxt]]$